Hardware Implementation of a Non-Coherent IR-UWB Receiver Synchronization Algorithm Targeting IEEE 802.15.6 Wireless BAN

Houcine Chougrani, Jean Schwoerer, Pierre-Henri Horrein, Amer Baghdadi

Hardware implementation of a non-coherent IR-UWB receiver synchronization algorithm targeting IEEE 802.15.6 wireless BAN

Houcine Chougrani*†, Jean Schwoerer*, Pierre-Henri Horrein†, Amer Baghdadi†
*Orange Labs, 28 Chemin du Vieux Chêne, 38240 Meylan, France
†Institut Mines-Telecom; Telecom Bretagne; Lab-STICC, Technopôle Brest-Iroise, 29238 Brest, France
Email: {houcine.chougrani,jean.schwoerer}@orange.com, {ph.horrein,amer.baghdadi}@telecom-bretagne.eu

***Abstract*—Synchronization acquisition is one of the main challenges for practical and efficient implementations of impulse radio ultra wideband (IR-UWB) receivers. This is particularly true in the context of the recently adopted IEEE 802.15.6 standard for wireless body area networks (BAN). Targeting energy-efficient non-coherent detectors, this paper presents a low-complexity hardware implementation of an efficient standard-compliant synchronization algorithm. The proposed architecture is described, together with performance and FPGA implementation results. A sub-optimal estimator of path selection and recombining is also proposed in the presented solution to improve the sensitivity of the receiver. Obtained results constitute a reference in this domain where the available literature is rather scarce.**

I. Introduction

The last few years have seen an increased interest in the research of practical and efficient receiver implementations for impulse radio ultra wideband (IR-UWB). One of the main recent drivers for this interest is the introduction in 2012 of the IEEE 802.15.6 standard for wireless body area networks (BAN). A very wide range of applications is foreseen as the standard targets short-range wireless devices for in-body, on-body, and around-the-body communications. Besides the very significant medical and healthcare application domain, the standard offers a huge opportunity for non-medical applications [1] belonging to a variety of fields including personal audio or video, gaming, entertainment, wearable computing, ambient intelligence, and many others. Therefore, these different application domains and the related communication scenarios lead to different technical requirements which must be met:

- Very low power consumption as some applications require devices with battery life of several months or even several years.
- Minimal short range.
- Variable data rate.
- Small factor form allowing portability of BAN devices.
- Robust communication quality between the BAN devices.

In fact, the IEEE 802.15.6 standard defines Physical (PHY) and Medium Access Control (MAC) Layers [2]. Three PHY are proposed: (1) body channel communication for signal propagation on the skin surface, (2) narrowband PHY mainly for healthcare applications using the various available license free band, and (3) ultra wideband (UWB) PHY able to address higher data rates. In this paper we focus on the impulse radio ultra wideband (IR-UWB) PHY, and more particularly on the synchronization issue at the receiver and its practical hardware implementation. Achieving an accurate synchronization is a major challenge in IR-UWB systems and a key factor to ensure reliable communications. Even a slight misalignment in the order of nanoseconds can severely degrade the system performance [3], [4], [5]. A recent overview of existing synchronization algorithms for IR-UWB systems is available here [5]. This overview paper analyses in particular the performance of few relevant recent synchronization algorithms: correlation based timing acquisition proposed in [6], orthogonal code matching based method in [7] and energy detection based method presented in [8]. As analysed in [9], state-of-the-art sliding correlation schemes have shown their efficiency in terms of timing retrieval accuracy but exhibit strong limitations in terms of power consumption and elapsed time to acquire a very fine common time base between emitter and receiver. The literature is rather scarce on non-coherent receivers synchronization acquisition [9], and particularly with a perspective of practical hardware implementations.

In this context, the new frame structure defined in the IEEE 802.15.6 standard for IR-UWB PHY offers new opportunities for efficient synchronization schemes. The specified synchronization header (SHR) integrates specific short 63-bit Kasami sequences which present good cross correlation properties (coexistence of BANs) and good autocorrelation properties for accurate synchronization. Considering non-coherent receivers, which are known to offer significant savings in complexity and in energy-per-bit [10][11] over their coherent counterparts, a recent work has proposed a new standard-compliant synchronization technique [12].

In this paper, we consider this recent synchronization technique and we present an efficient low-complexity hardware implementation that offers good opportunities for integration in low power BAN devices. The proposed architecture is described, together with performance and FPGA implementation results. Furthermore, a sub-optimal estimator of path selection and recombining is also implemented in the presented solution to improve the sensitivity of the receiver.

The rest of the paper is organized as follows. Section II gives a general overview of the transmitted frame structure, the IR-UWB non-coherent receiver, the considered synchronization algorithm as well as the proposed path selection and recombining estimator. Section III describes in details the proposed hardware architecture for the synchronization algorithm and for the path selection and recombining estimator. Section IV presents the results related to the synchronization success rate of the considered algorithm and the proposed FPGA hardware implementation. Finally, conclusions are drawn in Section V.

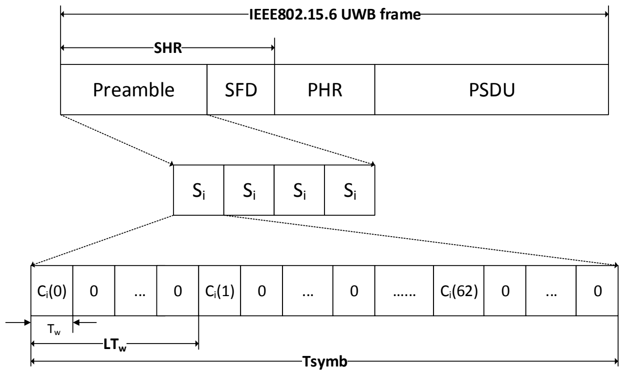


Fig. 1. UWB PHY frame format

II. SYNCHRONIZATION ALGORITHM

In this section, an overview of the transmitting frame and receiver analog front-end is presented. The synchronization algorithm and the proposed path selector are also described.

A. Transmitted frame

The UWB PHY frame format specified in the IEEE 802.15.6 standard [2] is composed of a Synchronization Header (SHR) used to acquire the synchronization, a Physical Header (PHR) containing information about the radio link (modulation, data rate, etc.) and Physical-layer Service Data Unit (PSDU) which is the payload of the frame. The SHR consists of the preamble, which is used for timing synchronization, packet detection, and carrier frequency offset recovery, and the start-of-frame delimiter (SFD), which is used for frame synchronization.

The preamble is built using a Kasami sequence of length 63. The standard defines eight different Kasami sequences which are named C_i for $i = 1, ..., 8$. The preamble consists of 4 repetitions of the symbol Si. Si is obtained by a Kasami sequence zero-padded by $L - 1$ zeros. Figure 1 illustrates the construction of the symbol S_i, where the zero-padding period is LT_w and T_w is the pulse waveform duration (T_w and L depend on the modulation employed) [13].

We consider in this paper an On-Off Keying modulation (OOK) and $LT_w = 64ns$. In this configuration, when sending a 1, the transmitter sends a pulse of $T_w{\sim}1ns$ and stays inactive for $63ns$. When sending a 0, the transmitter stays inactive for $64ns$.

B. IR-UWB non-coherent receiver

The structure of the receiver is illustrated in Figure 2. The analog front-end of the receiver is based on a non-coherent architecture. It is based on energy detection over a short integration period. It embeds a low-pass filter with short integration duration in the order of the pulse duration. The goal is to obtain the envelop of the received signal as a baseband pulse. This envelop is then compared to a predefined threshold in order to determine whether a pulse is being received or not.

The received signal can be represented using the output of the comparator. When the received signal is above the threshold, it gives a 1, otherwise it gives a 0. The presence of a pulse is then represented using this binary value.

It should be noted that using a comparator avoids the need for an analog-to-digital converter (ADC), which significantly decreases the complexity and the cost of the receiver. Consequently, such a low complexity structure allows simply to detect the presence/absence of the signal rather than a more accurate, yet high complexity, amplitude information (energy estimation) [12].

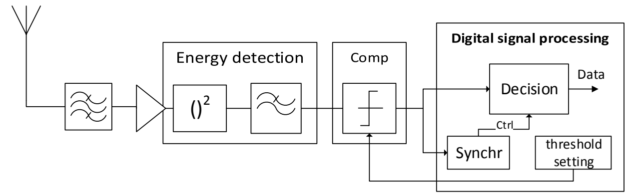


Fig. 2. Block diagram of the non-coherent receiver

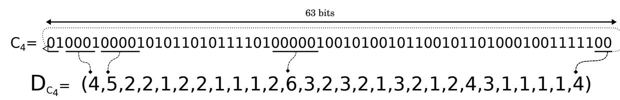


Fig. 3. Representations for Kasami sequence C_4

C. The synchronization algorithm

The considered synchronization algorithm has been recently proposed in [12] based on inter-pulse time interval detection and comparison. As presented above, the preamble symbol is based on Kasami sequence. The upper part of Figure 3 represents the fourth one (C_4) used in the considered link for illustration. The notion of slot is used here to simplify the comprehension of the algorithm, so one slot is a time quantity which represents the duration of one bit ($64ns$). When this sequence is sent, a pulse is transmitted in the first slot, then 3 slots stays unused, before having a new slot with a pulse. On the receiver side, this means that there is a time interval of four between the first two pulses. This sequence can thus be represented using a time interval representation, as shown in the lower part of Figure 3.

The synchronization algorithm is based on a finite-state machine (FSM) designed to check the correlation between the received symbol and the expected one. The FSM also allows to count the distance between two consecutive received pulses. The algorithm continues as long as the detected distances correspond to the expected ones, and restart when it is not the case. The synchronization is done when all the expected distances are identified.

In order to accommodate for the noise interfering with a transmission, the algorithm must be able to cope with spurious or missed pulses. The proposed FSM has been optimized to take into account a certain number of these erroneous pulses.

The main idea is that when a detected distance differs from the expected one, it is not discarded immediately. So, the comparisons to decide whether a distance matches the expected one is less strict, and can be made with the sum of expected distances (missed pulse) or the sum of received distances (spurious pulse). For example, if a distance $\{6\}$ is received, expected next distance is a $\{3\}$ as shown in the lower part of Figure 3. However, if the next pulse is missed, the next distance might be a $\{5\}$, which could be the sum of $\{3\}$ and $\{2\}$, the two distances following $\{6\}$. If a $\{5\}$ is received, the algorithm does not stop, and decide according to the next distance. On the opposite, the two next received distances could be $\{1\}$ and $\{2\}$, which should be discarded. However, the sum of received distances is 3, and this error could be caused by noise. This sequence can be kept if the following distances match the expected ones.

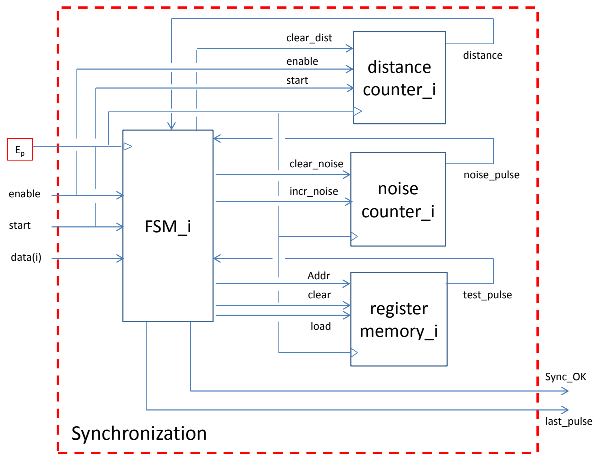


Fig. 4. Synchronization block

D. *Path selection and recombining estimator*

In the IR-UWB radio link, the transmitter sends a frame with the structure presented in Sub-section II-A. The zero-padding period is set to $LT_w = 64ns$, and OOK modulation is employed.

In the receiver part, the signal is detected for one pulse duration ($1ns$). This allows a multipath reception, which means that we can detect several paths and recombine them to increase the receiver performances. With zero-padding of $LT_w = 64ns$, we can detect up to 64 paths for a transmitted pulse.

To enable this multipath reception, we use a parallel synchronization scheme with multiple branches, where each branch embeds one FSM and processes $1ns$ detected signal. So to cover all possible path, 64 branches of synchronization are considered. Each FSM is responsible to detect a synchronization in its own branch. If this is the case, the detected signal on this branch is considered as a valid path.

The FSM optimization for error tolerance presented in Sub-section II-C can be used to sort the validated paths according to their estimated signal quality. When the energy of the path is high, it stays above the detection threshold during the whole synchronization symbol. As a result, the FSM converges without going through the optimization steps. This means that the FSM rapidly converges. In the same way, a path with relatively less energy does not reach the detection threshold during the whole transmission. In this case, one or more optimization steps are applied, leading to higher convergence time. Due to this difference in convergence time, the first detected paths can be safely used as the more powerful ones. This is a sub-optimal path selection, as the proposed low-complexity non-coherent receiver does not sort the paths whose FSMs converge at the same time given the lack of information on energy measurement. In this case, the first received path is considered as the most powerful.

III. Implementation

This section describes in details the proposed hardware architecture for the considered synchronization algorithm and for the proposed path selection and recombining estimator.

A. *Implementation of synchronization algorithm*

Figure 4 gives a general overview of the proposed architecture to implement the synchronization algorithm. The proposed architecture integrates the following main components: *distance_counter*, *noise_counter*, *FSM* and *register_memory*. The blocks *distance_counter* and *noise_counter* consist of simple counters. The first one (*distance_counter*) is used to compute the distances between the received pulses, whereas the second one(*noise_counter*) computes the number of spurious pulses when necessary. In the same way, the *FSM* block checks the different distances delivered by *distance_counter* component as described in Section II, and the *register_memory* component is used to check the identified pulses.

When the control part of the receiver detects a change in the channel, it activates the *Acq_Sync* signal to trigger the synchronization acquisition process. Once *Acq_Sync* is activated, the *synchronization* block starts taking into account the received data from the output of the comparator through the *data* signal.

When no pulse are detected (*data* = 0), *distance_counter* is incremented. Once a pulse is received (*data* = 1), the FSM reads the *distance_counter* value which represents the distance between two consecutive detections. At this state, if the distance corresponds to one of the expected distances, the FSM changes its state and resets the *distance_counter* by activating the *clear* signal. As a result, one pulse has been detected and the distance with the next pulse is begin computed. If the distance does not correspond to the expected one, the second pulse is considered as a noise, and *FSM* increments the *noise_counter* in order to verify the tolerate spurious pulses. As long as the tolerated spurious is not exceeded, the above process continues until the identification of expected distance. If it is not the case,the process should be restarted.

When a pulse j is identified, the register j of the *register_memory* component changes its value to 1, indicating that the j^{th} pulse was identified. The process described above is repeated until all the pulses are identified (*register_memory* is full). Once this happens, the FSM converges to the final state, activates the *Sync_ok* signal to indicate that the synchronization is acquired, and provides the position of the last detected pulse through the *last_pulse* signal. This *last_pulse* signal is used to represent the position of the current pulse, based on received distances. This position is given in binary coded decimal form.

B. *Implementation of path selection and recombining estimator*

Figure 5 presents the proposed architecture for the path selection and recombining estimator named *path_selector* block.

It consists of a serial-to-parallel converter (*deserializer* component) cascaded with 64 *synchronization* blocks. To allow processing of $1ns$ signal in each branch, the *deserializer* must work at high frequency, typically at $1GHz$. In order to ease the constraints, data processing is parallelized to reduce the working frequency of the other blocks.

The *deserializer* block is divided in two main parts:

- the clock managemet part, represented by the *clock_generator* block on the figure, is designed to provide proper clocks to the different parts of the system,

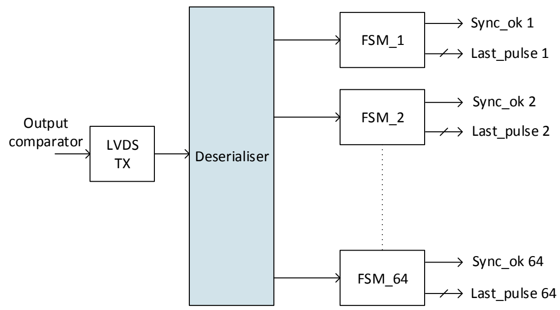


Fig. 5. Path selector block

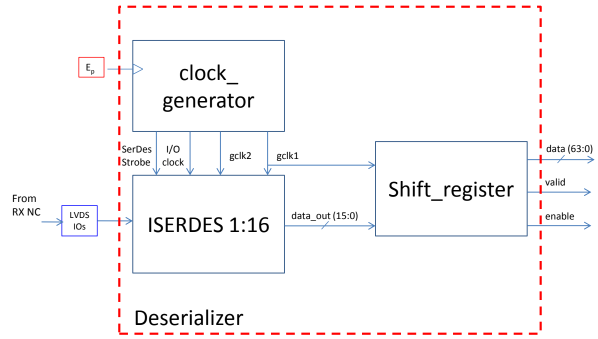


Fig. 6. Deserializer block

- the parallelization part, represented by blocks *ISERDES* and *shift_register*, is used to transform the serial input in a parallel processing unit.

In order to further describe this block, a target specific implementation is proposed here. The selected target is a Xilinx FPGA with high-speed inputs (RocketIO^{TM} GTP transceivers). The *clock_generator* component is based on one of the FPGA Digital Clock Management (DCM) unit.

The *ISERDES* block is provided by Xilinx, and designed to use hardware deserializers available in the FPGA. It disposes of SERDES ratios of 1:2, 1:3, and 1:4, where the SERDES ratio is defined as the ratio between the high speed I/O clock that is capturing data, and the slower internal global clock used for processing the parallel data [14]. However, the SERDES ratio can be extended to 1:8 when ISERDES is cascaded [15].

The digital receiver uses the same *Ep* clock as the analog front end. The *clock_generator* component uses this clock as input of its internal Phase-Locked Loop (PLL) to generate an input clock for the *ISERDES* block, with a frequency matching the required data rate (1 GHz). Represented by I/O clock in Figure 6, this high speed clock is then used to sample the received serial data on the input of *ISERDES* block. In the proposed non-coherent receiver, data is sent from the analog frontend to the digital part through an LVDS link, with no clock signal. Since the data rate is known, and the target is to sample the input data at each clock cycle, this allows the *ISERDES* block to be used as an analog-to-digital converter: the output of the comparator is sent as data signal, and the internally generated 1 GHz clock becomes a sampling clock.

To allow 1:16 SERDES ratio, an intermediate clock is generated by the *clock_generator* at 125 MHz (*gclk2*). From that clock, two ISERDES with 1:8 ratio are cascaded, forming a 16 bits word at the output of ISERDES (*data_out*) with a frequency of 62.5 MHz (*gclk1*). Given the 1:16 ratio, a 62.5 MHz clock for the 16 bits word is sufficient to allow a 1 GHz input data rate.

As mentioned above, 64 parallel data is required at the output of *deserializer* block, so a ratio of 1:16 is not sufficient. This last parallelization step is done by the *shift_register*, which takes as input the 16 bits words sent by *ISERDES* and outputs a 64 bits word with a data rate divided by 4. For performance reason, a new clock is not generated for this final output, but the *enable* signal is used to indicate the validity of the output word. The system clock for the digital receiver is thus *gclk1*, and 4 cycles are available to process each 64 bits data word.

At this stage, a simple connection of each output data of the *deserializer* block to a *synchronization* block is sufficient to complete the *path_selector* block as illustrated in Figure 5.

IV. Results

This section presents the results related to the synchronization software simulation of the considered algorithm and the proposed FPGA hardware implementation

A. Synchronization software simulation

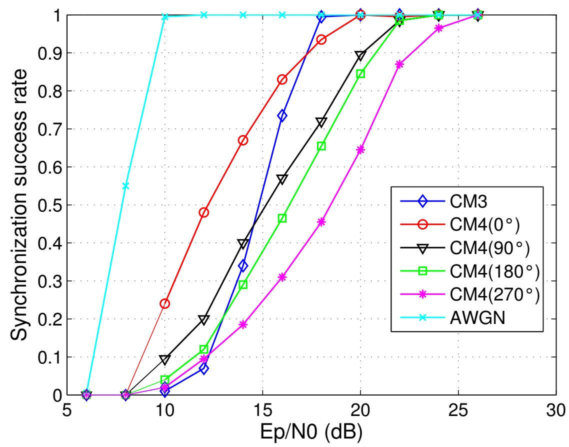
This considered synchronization algorithm [12] was implemented in a complete UWB BAN simulation environment developed in Matlab for the RUBY project[1]. Simulations were performed for additive white Gaussian noise (AWGN) and for CM3/CM4 channel models [16] which are defined by the IEEE 802.15.6 standard.

Figure 7(a) illustrates the synchronization success rate of the considered synchronization technique. As can be seen, the difference between all channels is not negligible. For the AWGN channel, the 100% synchronization success rate is obtained when Ep/N0 reaches 10 dB, while CM3/CM4 channel models obtain the same synchronization success rate starting from 17 dB. This can be explained by the multipath characteristic of the CM3/CM4 channel models. A difference between these channel models is also observed. The worst cases are CM4 for angles other than 0°.

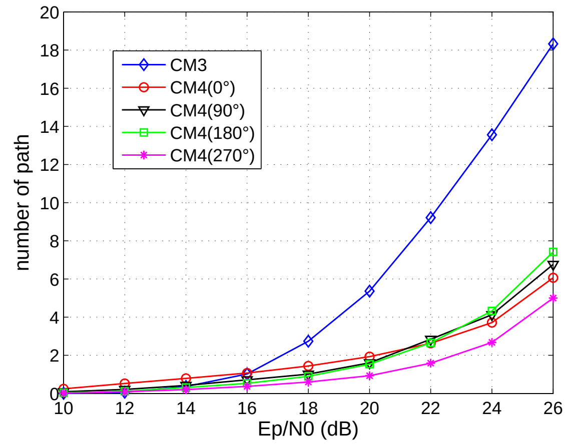
Figure 7(b) shows the detected paths for the different CM3/CM4 channel models. As we can see, the identified paths in CM3 is more important than the CM4. More than 18 paths are identified in the case of CM3 when Ep/N0 reach 26 dB, while the maximum identified paths is around 8 for CM4 at the same Ep/N0. CM4 (0°) is a Line-Of-Sight (LOS) channel, with a strong main path. CM3 may be either LOS or Non-LOS (NLOS), depending on whether a part of the body is located between devices. CM4 for angles other than 0°are either LOS or NLOS, depending on the position of the body. However, the distance between devices is much higher than for CM3, and the spreading of paths over time is more important, which explains why results are lower than CM3.

A comparison of our considered synchronization technique results with [17] results was carried out in [12]. This comparison revealed the effectiveness of our algorithm in term of Synchronization success rate as well as the time required to acquire the synchronization.

[1]Research project started in 2012 and funded in part by the French National Research Agency (ANR)

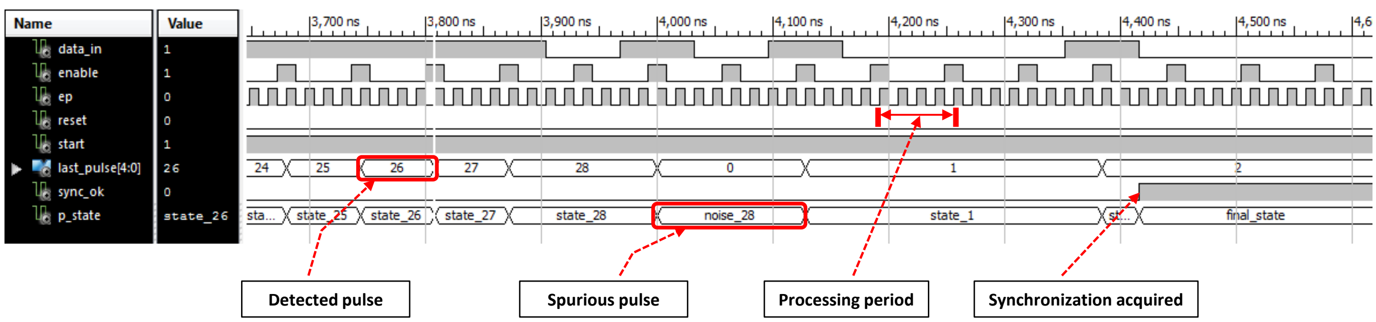


(a) Synchronization success rate

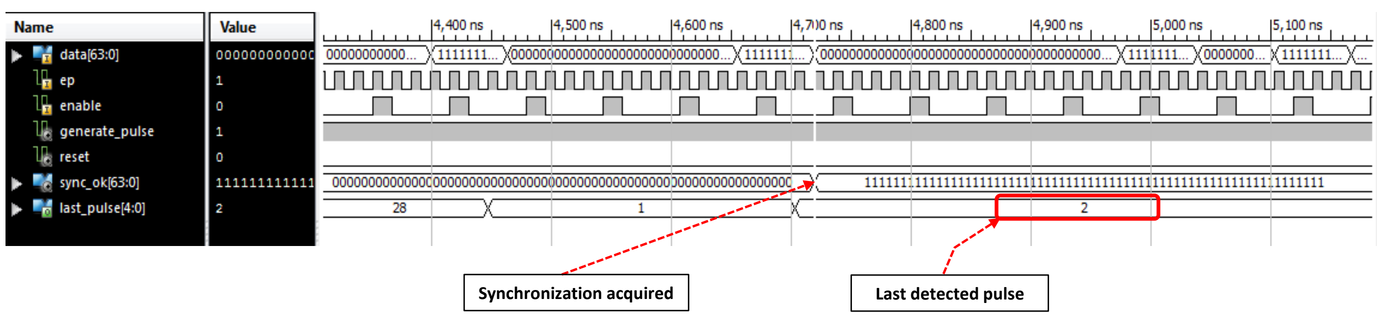


(b) number of detected path

Fig. 7. Simulation results of the proposed algorithm for various channel models



(a) *Synchronization* block



(b) *path_selector* block

Fig. 8. Example of signal waveform results of the implemented hardware synchronization architecture

B. FPGA hardware implementation

The architecture presented in Section III has been fully described in VHDL including the input interface with the Deserializer block (Figure 6). For testing purpose, a transmitter block has also been developed in VHDL to generate an impulse signal compliant with the standard frame structure as presented in Sub-section II-A. Besides the behavioral and post-synthesis simulations, on-board validation was performed using the development board AES-S6DEV-LX150T-G from Avnet which integrates a Xilinx Spartan 6 LX150T FPGA circuit. The selected FPGA circuit integrates 8 RocketIO^{TM} GTP transceivers able to operate at a serial bit rate of 3.125 Gb/s (as the integrated FPGA is of a -3 speed grade). Both the transmitter and the receiver have been implemented on this single FPGA. However, the outputs of the transmitter are sent to two LVDS output pins of the FPGA and then connected to two LVDS input pins of the receiver. For the on-board validation, Xilinx ChipScope Analyzer was used to probe the internal signals of the design and check the different implemented blocks. Different analysis have been conducted and the synchronization acquisition was verified with spurious and missed pulses. Figure 8 illustrates an example of signal waveform results.

The output signals as well as the transition states evolution of the *synchronization* block are presented in the upper part of Figure 8(a). Signals are presented in Section III. The Figure shows the end of a successful synchronization in

TABLE I
FPGA RESOURCE UTILIZATION

Logic Utilisation	*synchronization* block			*path_selector* block		
	Used	Available	Utilization	Used	Available	Utilization
Number of Slice Registers (Flip-Flops)	127	184304	0%	8276	184304	4%
Number of Slice LUTs	431	92152	0%	27056	92152	29%
Number of BUFG/BUFGCTRLs	1	16	6%	4	16	25%
Number of PLL_ADVs	0	6	0%	1	6	16%

the presence of a spurious pulse. The received sequence at this point is ”111101010001”, which is a noisy version of ”111100010001”. This matches the end (”111100”) and start (”010001”) of C_4 sequence. The processing period of 4 cycles can be seen between each *enable*. The FSM changes state each time a pulse is received, and the *last_pulse* signal is update with current pulse position. When the spurious pulse is received, it does not match an expected pulse, and the FSM goes through a *noise* state. Finally, when the last pulse is received, the sequence is recognized, the *Sync_ok* signal is set to 1 to indicate that synchronization is acquired. The *last_pulse* signal thus represent the position of this pulse. The results for *path_selector* block are shown on Figure 8(b). The *sync_ok* signal is the concatenation of all *sync_ok* from the 64 *synchronization* blocks. This shows the results for the same sequence duplicated on each branch, and it can be seen that synchronization is found on all branches, indicating that the block works as expected.

Table I summarizes the required hardware resources to implement the *synchronization* and *path_selector* blocks. These results illustrate the hardware efficiency of the proposed architecture. A single *synchronization* block requires very low resources utilization: 127 Flip-Flops and 431 Look-Up Tables (LUT). Moreover, it can be noticed the absence of multipliers (DSP's resources) in the resources occupation.

Regarding the *path_selector* block, the overall resources utilization in terms of Flip-Flops and LUTs is almost multiplied by 64. This is necessary in order to analyze the 64 possible paths and to allow a robust multipath detection. The use of the PLL in the *path_selector* block leads to use more buffers in order to drive the different generated clock. This explain the difference of the used buffers between *synchronization* block (one buffer) and *path_selector* block (four buffers). Even with this advanced feature, the resulted complexity is still reasonably low: about 8K Flip-Flops ($\sim$4% of the available resources) and 27K LUTs ($\sim$30% of the available resources). Furthermore, this additional complexity is compensated by the increased synchronization performances when the multiple paths are recombining to take a decision. Thus, the proposed low-complexity synchronization architecture offers good opportunities for integration in low power BAN devices.

V. CONCLUSION

In this paper, a low-complexity hardware implementation of an efficient synchronization algorithm for non-coherent IR-UWB receivers is presented. The proposed architecture is compliant with the recently adopted IEEE 802.15.6 standard for BANs. It is based on the use of an optimized FSM with appropriate counters and registers to match the particular structure of the synchronization header based on short 63-bit Kasami sequences. The presented solution integrates a sub-optimal estimator of path selection and recombining to improve the sensitivity of the receiver. The efficient synchronization algorithm enables a effectiveness synchronization success rate in BAN channel models. FPGA implementation results show a reasonably low resources occupation (about 27K LUT and 8K FF) which offers good opportunities for integration in low power BAN devices. Further optimizations are possible by making use of the available 4 cycles, which could further reduce required resources. Obtained results constitute a reference in this domain where the available literature is rather scarce.

REFERENCES

[1] S. Drude, “Requirements and application scenarios for body area networks,” in *Mobile and Wireless Communications Summit, 2007. 16th IST*, 1-5 July 2007.

[2] *Part 15.6: Wireless Body Area Networks*, IEEE Standard for Local and metropolitan area networks Std., 29 February 2012.

[3] Z. Tian and G. Giannakis, “BER sensitivity to mistiming in ultra-wideband impulse Radios-part I: nonrandom channels,” *IEEE Transactions on Signal Processing*, vol. 53, no. 4, pp. 1550–1560, 2005.

[4] N. He and C. Tepedelenlioglu, “Performance analysis of non-coherent UWB receivers at different synchronization levels,” *IEEE Transactions on Wireless Communications*, vol. 5, no. 6, pp. 1266–1273, 2006.

[5] R. Akbar and E. Radoi, “An overview of synchronization algorithms for IR-UWB systems,” in *Proc. of the International Conference on Computing, Networking and Communications (ICNC)*, 2012, pp. 573–577.

[6] L. Yang and G. Giannakis, “Timing ultra-wideband signals with dirty templates,” *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1952–1963, 2005.

[7] Y. Ying, M. Ghogho, and A. Swami, “Code-Assisted Synchronization for UWB-IR Systems: Algorithms and Analysis,” *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 5169–5180, 2008.

[8] X. Luo and G. Giannakis, “Low-complexity blind synchronization and demodulation for (ultra-)wideband multi-user ad hoc access,” *IEEE Transactions on Wireless Communications*, vol. 5, no. 7, pp. 1930–1941, 2006.

[9] B. Miscopein and J. Schwoerer, “Low Complexity Synchronization Algorithm for Non-Coherent UWB-IR Receivers,” in *Proc. of the IEEE 65th Vehicular Technology Conference (VTC2007-Spring)*, April 2007, pp. 2344–2348.

[10] F. S. Lee and A. P. Chandrakasan, “A 2.5 nj/bit 0.65 v pulsed uwb receiver in 90 nm cmos,” *Solid-State Circuits, IEEE Journal of*, vol. 42 Issue:12, pp. 2851 – 2859, Dec 2007.

[11] Y. Gao, Y. Zheng, and C.-H. Heng, “Low-power cmos rf front-end for non-coherent ir-uwb receiver,” in *Solid-State Circuits Conference, 2008. ESSCIRC 2008. 34th European*, 15-19 Sept. 2008.

[12] H. Chougrani, J. Schwoerer, P.-H. Horrein, and A. Baghdadi, “Efficient Synchronization Technique for Non-coherent IR-UWB Receiver Targeting IEEE 802.15.6 Wireless BAN,” in *Proceedings of the 8th International Conference on Body Area Networks (BodyNets)*, 2013, pp. 181–184.

[13] *Part 15.6: Wireless Body Area Networks*, IEEE Standard for Local and metropolitan area networks Std., 2012.

[14] “Spartan-6 fpga selectio resources,” XILINX, Tech. Rep., February 14, 2014.

[15] N. Sawyer, “Source-synchronous serialization and deserialization (up to 1050 mb/s),” XILINX, Tech. Rep., June 3, 2010.

[16] K. Y. Yazdandoost and K. Sayrafian-Pour, “Channel Model for Body Area Network (BAN),” *IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)*, vol. Document IEEE802.15-08-0780-05-0006, 2009.

[17] K. B. et al, “Etri & samsung phy proposal to 802.15.6,” IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs), Tech. Rep., May 2009.